

# AllPurdue

## Design Document

### Team 10

- Antony Ni
- Andrew Ni
- Yash Narain Agarwal
- Pramey Kabra
- Matthew Stroup
- Rasesh Ramadesikan

## Index

● <b>Purpose</b>	<b>3</b>
○ Functional Requirements	3
○ Non-Functional Requirements	6
● <b>Design Outline</b>	<b>7</b>
○ High Level Overview	7
○ UML Diagram	8
● <b>Design Issues</b>	<b>9</b>
○ Functional Issues	9
○ Non-Functional Issues	10
● <b>Design Details</b>	<b>12</b>
○ Class Diagram	12
○ Descriptions of Classes and Interaction between Classes	12
○ Sequence Diagrams	15
○ UI Mockups	26

## Purpose

There is not a single centralized Purdue service to help boilermakers in their search for places to eat, study, or live. Such a platform has become a necessity since the currently existing resources are extremely scattered and not at all geared toward giving the Purdue community ideal recommendations and reviews. Our application targets all the Purdue students, alumni, and visitors in the West Lafayette area, who are looking for honest suggestions and want to share their unique experiences at and around the vibrant campus.

Individual review services like Yelp and Google exist where you can look up places and reviews, but there is no centralized platform that is tailored for Boilermakers; AllPurdue solves this problem by providing a convenient, streamlined platform acting as a one-stop service, optimized for the people and visitors of Purdue.

Whether it be a food place, a residence hall, or a study spot at Purdue, AllPurdue would have a wide variety of locations in different categories containing ratings and reviews made by Boilermakers, as well as visitors and locals. Unlike our competitors, AllPurdue users can also create blog posts talking about their experiences at various places, helping others in their search for a place to eat, live or study.

While ratings and reviews of the places are the central features for services like this, AllPurdue does not only allow users to do that, but a lot more including creating blog posts. Through blog posts, users can write about their experiences in more detail, talk about their favorite places, etc., which also helps the other users in making their decisions, as they get to know a lot more than just quantitative information, about the different places.

## Functional Requirements

### 1. User Account and Login :

As a user,

- I would like to be able to register for an AllPurdue account.
- I would like to be able to log in and manage my AllPurdue account.
- I would like to have a secure and fast login process.
- I would like my password to be reset if I forget it by getting a verification email.
- I would like to, as a Purdue student, staff, faculty, or alumni, have my account auto-verified when registering with my Purdue email.

### 2. Searching and Browsing:

As a user,

- I would like to be able to search for any place at Purdue from the home page.
- I would like to filter and sort results by the average rating.
- I would like to see the top-rated places for each category so that I can find a suitable place quickly.

- I would like to filter places by the tag(s) associated with the locations.
- I would like to be able to filter my searches by the number of ratings so I can find out the most popular places (higher number of ratings) or discover hidden gems (lesser number of ratings).
- I would like to be able to see popular and trending locations currently, based on the most number of ratings received by a place in a recent time period, so I can stay up-to-date.
- I would like to be able to filter my searches by places that have offerings within a price range (\$, \$\$, \$\$\$, \$\$\$\$) so that I can find affordable places to eat and live.
- I would like to be able to filter my searches for food/drink locations that also offer complimentary wi-fi so that I would be able to have a study session while also satisfying my appetite.
- I would like to be able to filter my searches by a numerical value (average rating, number of reviews) within each category.
- I want to be able to view locations with the most study resources, such as printing and computer facilities.
- I would like to be able to see individual categories (food, housing, study spots, and blog posts) with separate tabs for each.
- I want to be able to view at least a few photos of each location so that I can get a better understanding of what a place looks like.
- I would like to request a location to be added to AllPurdue.

### 3. **Map Features:**

As a user,

- I would like to see an interactive map of all places on AllPurdue.
- I would like to filter results on the map using the distance from my current location.
- I would like to filter results on the map using the category.
- I would like to be able to get directions to all places on AllPurdue.

### 4. **Interaction with the page for a specific location:**

As a business owner, I want to be able to claim my business on AllPurdue, so that I can request updated information about my business.

As a user,

- I would like to be able to favorite and unfavorite locations so that I can quickly view them in the future.
- I would like to be able to receive notifications about my suggestions and any changes made by business owners of my favorite places.
- I would like to be able to rate all places on AllPurdue on a scale of 1 to 5 stars.
- I would like to write reviews for all places on AllPurdue.
- I would like to be able to delete my review(s).

- I would like to be able to see the average rating when on the page for a particular location.
- I would like to be able to see when a location was added to AllPurdue so that I can see what people think about it over time.
- I would like to be able to see the basic details about a place, such as hours of operation and address(es), so that I can plan my visit(s) accordingly.
- I would like to be able to see reviews of a particular location by other users and like reviews that I think are helpful.
- I would like to be able to filter reviews by those only made by verified Purdue users so that I can get a better understanding of what other students or faculty members think of a particular place.
- I would like to be able to view reviews based on time posted (newest/oldest).
- I would like to view reviews by rating given with the review.
- I would like to view reviews based on the number of likes.
- I would like to be able to see recommendations for similar places for all places on AllPurdue.
- I would like to be able to share all places on AllPurdue with my friends using email, text, or other services.
- I would like to be able to suggest details and tags for a place.

#### 5. **Blog Posts:**

As a user,

- I would like to be able to view, save, and like other users' blog posts.
- I would like to be able to sort blog posts by the most likes or recent posts.
- I would like to be able to filter blog posts by the type of location it talks about as well as using tags.
- I would like to be able to create blog posts so that I can share my experience(s) in detail for the benefit of other users as well as for my own future reference.
- I want to be able to add photos to my blog posts to make them more comprehensive.
- I would like to be able to delete my blog post(s).

#### 6. **Managing the Platform:**

As an administrator,

- I want to be able to manage the list of all locations on AllPurdue.
- I want to be able to add new locations and/or tags for specific locations based on user suggestions whenever relevant.
- I want to be able to update information about a location upon the requests of verified business owners.
- I want to be able to remove reviews and blog posts that may promote harmful conduct.

## **Non-Functional Requirements**

### **Architecture and Performance**

Our full-stack application will be separated into a frontend and backend, allowing us to develop in two teams, a frontend team and a backend team, each consisting of three developers. This effectively makes the development process quicker and easier to manage, as each team will be working on features of the application relevant to their team (i.e. the frontend team will focus on the implementation of frontend features, and backend team will focus on backend features). AllPurdue will have 99% uptime being hosted on Heroku.

The frontend of the application will be developed using ReactJS. We will be sending requests to the backend API using Axios (a React library) to get necessary data in < 500ms, such as the reviews and ratings of places on and around campus. Additionally, Google Maps API will be utilized to display an in app map of the various locations of places.

The backend of the application will be developed using NodeJS, Express, and MongoDB. NodeJS and Express will be utilized to create a RESTful API and MongoDB will be used to create a scalable database containing all of the data.

### **Usability**

The user interface will be simple and easy-to-use for displaying reviews and ratings. The interface will be intuitive, with easily recognizable icons and buttons, and a logical organization of information. Clutter will be reduced by organizing the information using a navbar to display various places by category. The user interface will be fully responsive, providing an optimal viewing experience on all devices, including desktop computers, laptops, tablets, and smartphones, making this application accessible across all devices.

### **Security**

Users can submit incident reports about any issues that may be occurring in the application. We can manually verify and remove any ratings, reviews, and blog posts if necessary. All API endpoints will be authenticated, allowing every request to be verified. The MongoDB database will be secured with MongoDB's built in tools: authentication, authorization, encryption, auditing, and network isolation.

### **Hosting/Deployment**

The application will be deployed and hosted using Heroku which allows us to easily deploy, run, and manage the application. The app will be configured to automatically deploy new code when changes are pushed to the GitHub repository. Before deployment, all testing for the frontend and backend will be done in a local environment.

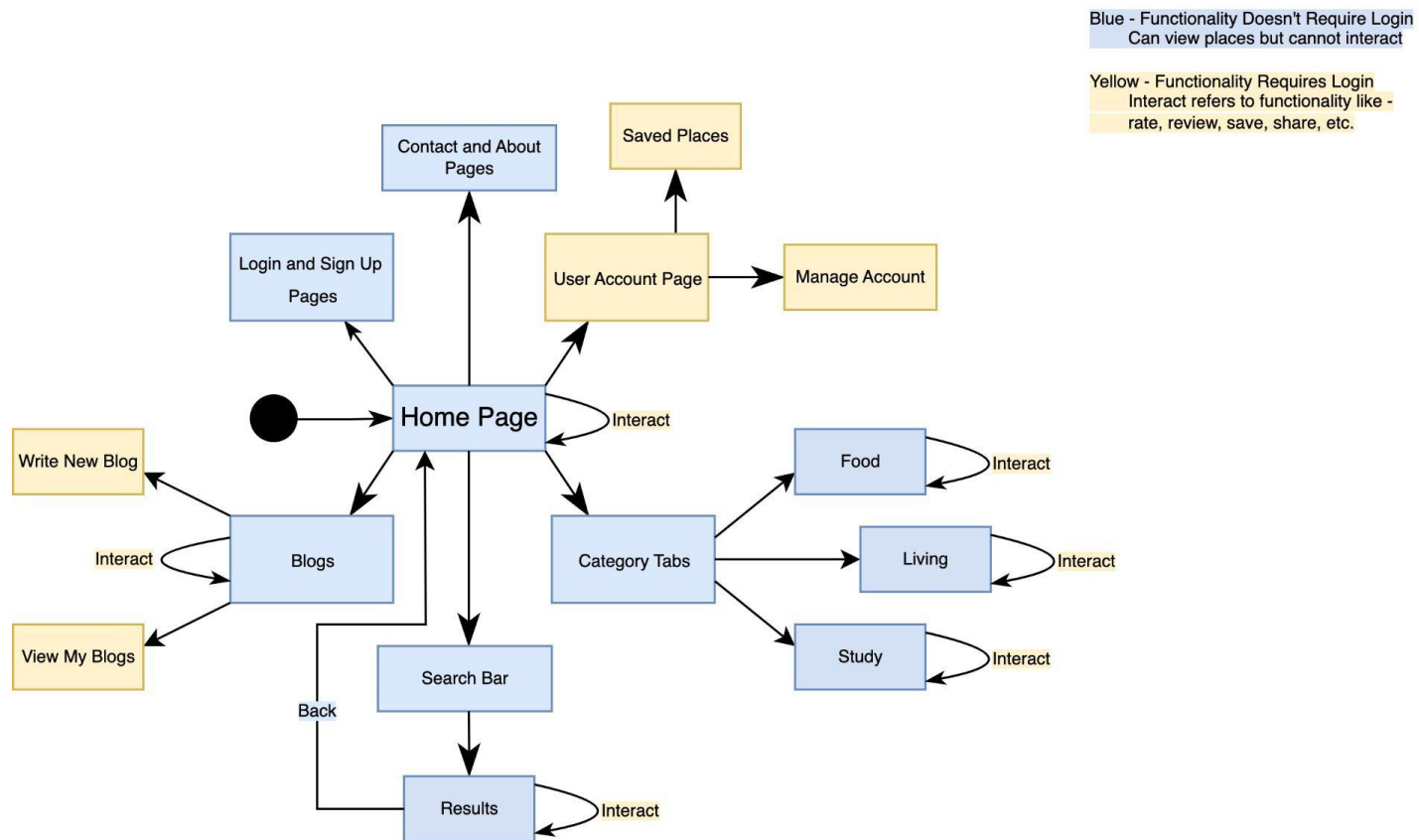
## Design Outline

### High Level Overview

This project is a web application that allows a user to view places at and around Purdue, rate and review them, and create blog posts to share their experiences. The application will be created using a sophisticated client-server architecture, allowing multiple clients to send requests to a central server, which will further communicate with a database to form queries and deliver responses. Furthermore, the server will implement the Model View Controller paradigm. The system, using the MERN stack base, will support multiple simultaneous user connections and also support individual account logging through the SAML framework.

1. Client
  - a. Client will access the frontend forming the entry point of the user interface.
  - b. Client can send CRUD requests to the backend and display the incoming responses in a serialized manner.
  - c. Client will display a list of features such as navigation tabs, search bar, login button, and a clean UI for blogging and ratings/reviews.
2. Server
  - a. Server will handle the CRUD requests from the client.
  - b. Server validates the incoming requests, serializes them, sends the queries to add, modify, or fetch data from the database.
  - c. Server generates the appropriate responses and sends them to the clients.
  - d. Server can also make API requests to external APIs such as Google Maps and fetch appropriate information.
3. Database
  - a. A non-relational (NoSQL) database that will store all the user data, right from the account information to ratings/reviews given to blog post content.
  - b. It will respond to incoming CRUD and Fetch queries from the server and send the appropriate responses.

## UML Diagram



This diagram shows the UML model for the entire application. It illustrates the pages and features that can be used by the users. Upon first entering the web application, the user lands on the Home Page. The home page will show an array of places that have been recently reviewed or rated and recent blog posts by users. The users can view detailed information about all the places by clicking on them and view ratings and reviews by other users. The home page also has multiple other pages linked on it such as - Sign-Up and Login, Contact and About Us, Manage User Account, Blogs, and Categories. The user can navigate to any of these pages to perform different actions. Some functionality on the application requires that the user have an account and is signed-in to the application. The users need to be signed-in to the application to perform certain actions such as rate, review, and save places, and to write, like, or comment on blog posts. These actions by the user are “interactions” with the places or blog posts.



## Design Issues

### Functional Issues

1. Issue: Do we need users to create a profile and login to use AllPurdue?

Options:

- a. Allow users to login using Google authentication.
- b. Have users create an account with their email and a login password.
- c. Use the service without any form of login or profile creation.

Decision: We decided that users would be able to use the service to browse and look up locations offered without creating any user account. But, if a user wanted to rate or review a place, add locations to their favorites, or make blog posts on our platform, we decided that it would require for them to create an account using their email and a login password. We also decided that to offer a layer of authenticity to the reviews/blog posts of users who use their Purdue email to create their accounts will be authenticated as a verified user.

2. Issue: How do we define tags that can be attached to each location?

Options:

- a. Have a predefined set of tags that cannot be modified.
- b. Allow users to customize and add tags to a location as they wished.
- c. Have a set of basic predefined tags that can be modified based on requests by users.

Decision: We decided to go with option c. This is because having a predefined set of tags establishes a centralized system which allows users to search for and filter locations, as well as suggest locations to users, without them having to manually add tags, which would add to the convenience of the user as well as improve the overall experience as personalized tags could vary from user to user too. However, we do want users to be able to suggest tags which they think is a crucial aspect of certain locations, which may be added upon verification by an administrator, adding to the overall user experience with AllPurdue.

3. Issue: How should we display the different types of locations and features of our platform?

Options:

- a. Have a tab for blog posts and display all types of locations in one tab with options to search for any particular location and filter the locations shown.
- b. Have a separate tab for blog posts and for each type of location on the service with options to filter/sort the entries in each tab, while having a centralized search option.

Decision: We decided to go with option b, because it would make the platform a lot less cluttered and also allow for a more organized experience, allowing users to look for

exactly what they want by having separate tabs for food and drink locations, residence halls, study spots, and blog posts, instead of showing everything available at one go. We decided to keep the centralized search feature so that if a user is looking for a place, they would be able to view all relevant entries available on our service.

4. Issue: What kind of interaction should we allow users to have with a location.

Options:

- a. Allow users to just rate the locations on the platform.
- b. Allow users to rate and review the locations on the platform and
  - i. also allow users to comment on and like/dislike other users' reviews or,
  - ii. also allow users to like reviews by other users.

Decision: We decided to not restrict users to just rating a location as other users would be missing out on valuable insight in the form of reviews. However adding comments and the option to like/dislike reviews would take away from the streamlined nature of the platform. Although, allowing users to just like other reviews would be helpful in boosting reviews that more users have found insightful and could be valuable. Our platform allows users to create blog posts in which they can offer more elaborate thoughts about places and possibly reasonings for their reviews, eliminating the need for comment threads on reviews.

## Non-Functional Issues

1. What frontend framework or language will we use?

- a. HTML and JavaScript
- b. Angular
- c. ReactJS
- d. Vue.js

Decision: We chose React as our frontend framework because of its power and simplicity. Many members on our team do not have experience with multiple frontend frameworks, so React is a great choice so every developer can learn and adapt quickly. Along with this, we will be using Axios, a React library, to handle backend API requests, which will make retrieving various data simple.

2. What backend framework or language will we use?

- a. Fastify
- b. Express.JS
- c. Spring Boot
- d. C++

Decision: We chose to develop our backend using both Node and Express to complement the frontend being built using React. Using both Node and React together has the potential to create a very powerful application which runs extremely quickly with

convenient compatibility functions. Adding Express on top of this will help create a RESTful API and manage data.

3. Which type of database will we use?

- a. MySQL (Relational/SQL)
- b. MongoDB (NoSQL)

Decision: We chose to use MongoDB as the database for our project due to its built-in tools and easy accessibility with Node.js. MongoDB has built in security tools including authentication, authorization, encryption, auditing, and network isolation, meaning our data will be extremely secure. MongoDB is also open source, and extremely popular, meaning online documentation and assistance will be plenty.

4. Which cloud platform will we use to host?

- a. Microsoft Azure
- b. Amazon Web Services
- c. Heroku
- d. Linux VPS

Decision: We chose to use Heroku as a hosting service. With Heroku's PaaS service, we will easily be able to deploy, run, and manage the application for free with little to no downtime. We will be able to scale the application easily without having to worry about managing the underlying infrastructure, overall reducing the time dealt with our hosting platform, instead focusing more on our architecture and development.

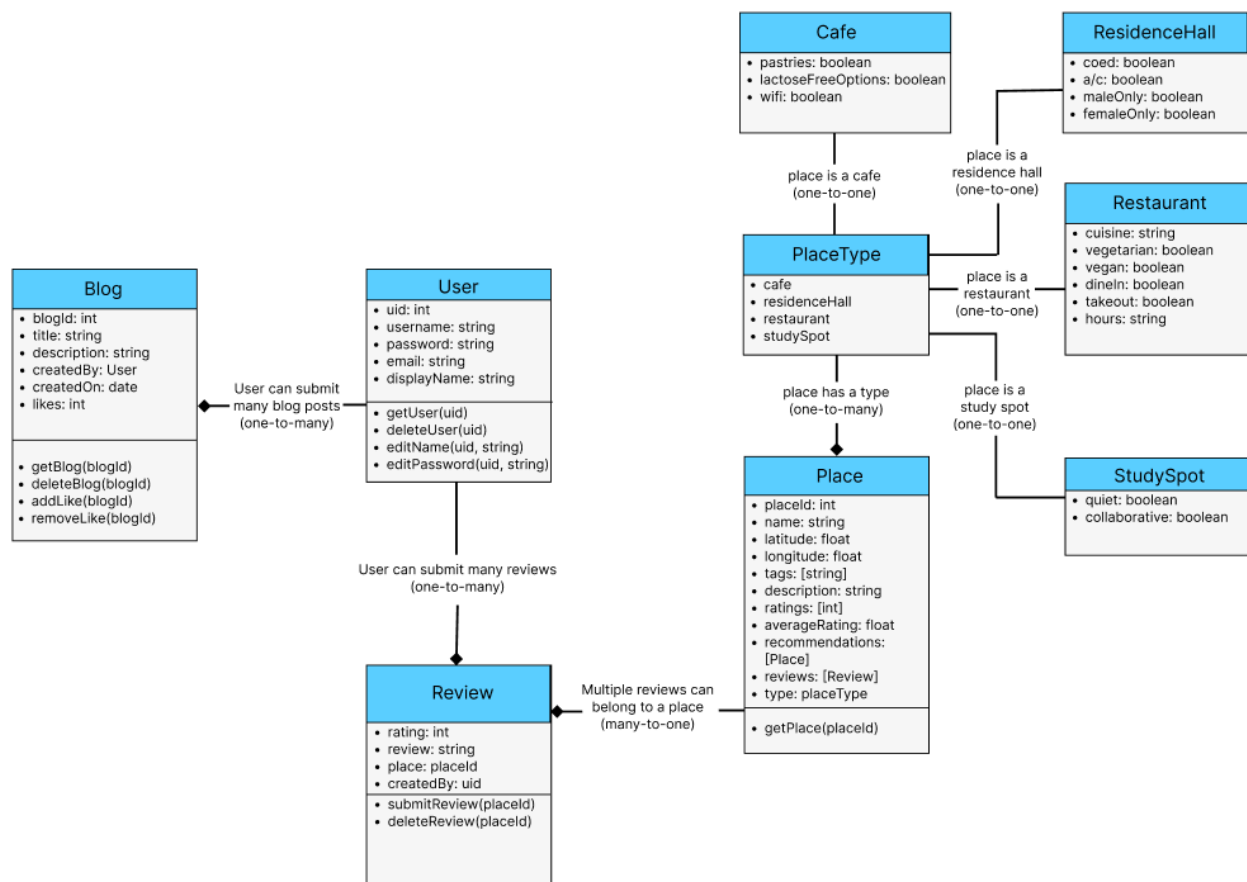
5. Which tool will we use to handle backend API requests?

- a. The native JS Fetch API
- b. jQuery AJAX
- c. Axios
- d. got

Decision: We chose to use Axios to handle our HTTP requests. This will allow us to make requests not only to a browser but also to a Node.js server. With our choice of ReactJS, we will be able to make requests using Axios, and have React display the results in the UI. Axios is a very simple to use library that is very well-documented with great quality of life tools such as automatic transforming for JSON data, making it the clear choice.

## Design Details

### Class Diagram



### Descriptions of Classes and Interaction between Classes

The schema models (MongoDB) are designed based on the objects in our application. The schema models list all of the properties of each object.

- **User**

- User schema is created with every new signup
- A unique user id will be assigned to each user
- Each user will have the username and/or email and password they used to sign up to allow for future logins
- An email is attached to the user to allow for account authentication and password resets
- Each user will have a display name. The display name will be shown on blogs and reviews posted by the user.
- Functions:
  - `getUser(uid)` - retrieves a user's data given their uid
  - `deleteUser(uid)` - deletes a user given their uid

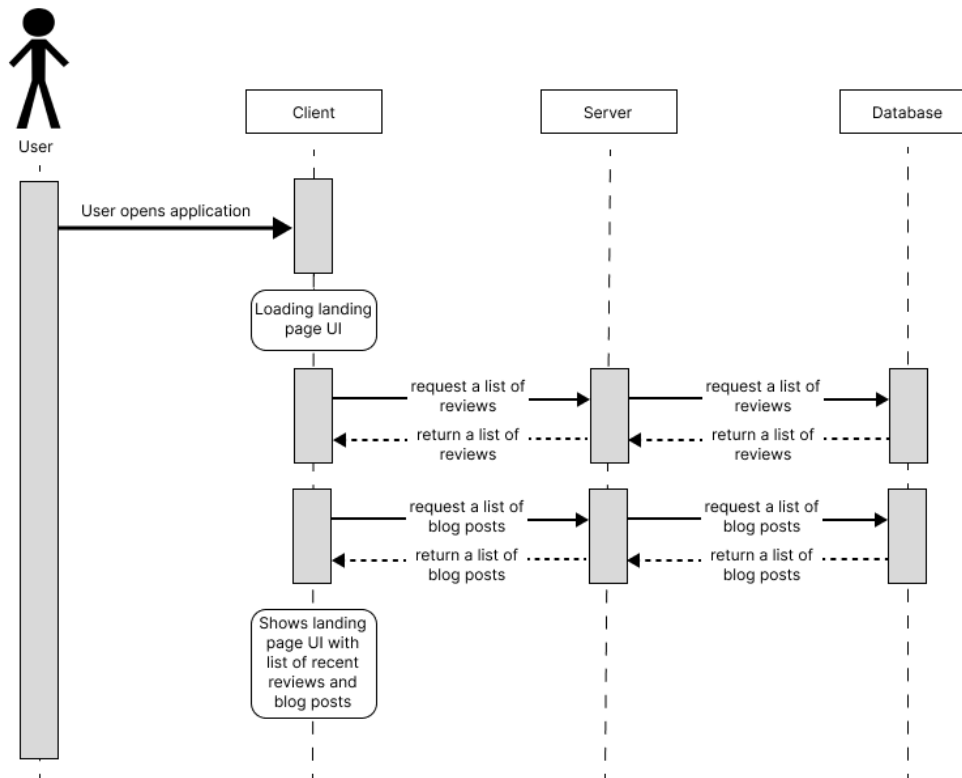
- editName(uid, string) - replaces a user's existing name with the input string
- editPassword(uid, string) - replaces a user's existing password with the input string
- **Blog**
  - Blog object is created when a user creates a blog
  - Each blog will be assigned a unique blogId
  - Each blog will have a title and description (contents of the blog)
  - The user who created the blog will have their uid stored with it
  - The date the blog post was created will be stored
  - The number of upvotes/likes will be stored and displayed
  - The creator of the blog post will be able to remove the blog post
  - Functions:
    - getBlog(blogId) - retrieves a blog post with a blogId
    - deleteBlog(blogId) - deletes a blog post with a blogId
    - addLike(blogId) - adds 1 to a blog's "likes"
    - removeLike(blogId) - removes a like from a blog post
- **Review**
  - A review object is created when a user submits a review for a place
  - Each review will include a rating (1-5 stars) and the review itself (a brief explanation for the rating)
  - Each review will be associated with the place in which it was submitted for (placeId)
  - Each review will have the uid of the user who submitted the review
  - The creator of the review and administrators will be able to delete the submitted review
  - Functions:
    - submitReview(placeId) - adds a review to a place given the place's placeId
    - deleteReview(placeId) - deletes an existing review from a place given the place's placeId
- **Place**
  - Represents a point of interest
  - Each place will be assigned a unique id (placeId)
  - Each place will have a name and description
  - Each place will have a longitude and latitude value assigned according to their location
  - Each place will have tags to better categorize them such type of cuisine and male or female residence halls
  - Each place will have an option to submit a Review

- Each place will have their individual ratings displayed and the overall average of the ratings
- Each place will have recommendations assigned to them based on their tags
- Each place will have written reviews
- Each place will be assigned a place type (placeType) depending on whether it is a restaurant, cafe, study spot, or residence hall
- Functions:
  - `getPlace(placeId)` - retrieves a place's data given the place's placeId
- **PlaceType**
  - An identifier to categorize a place
  - placeType can be cafe, residenceHall, restaurant, or studySpot
  - **Cafe**
    - Each cafe has a boolean to indicate whether pastries are available
    - Each cafe has a boolean to indicate whether lactoseFreeOptions are available
    - Each cafe has a boolean to indicate whether wifi is available
  - **Restaurant**
    - Each restaurant has a string variable indicating the type of cuisine offered at the restaurant
    - Each restaurant has a boolean to indicate whether vegetarian options are available
    - Each restaurant has a boolean to indicate whether vegan options are available
    - Each restaurant has a boolean to indicate whether dining in is available
    - Each restaurant has a boolean to indicate whether takeout is available
    - Each restaurant has an string variable indicating restaurant hours
  - **ResidenceHall**
    - Each residenceHall has a boolean to indicate coed living
    - Each residenceHall has a boolean to indicate whether a/c is available
    - Each residenceHall has a boolean to indicate maleOnly
    - Each residenceHall has a boolean to indicate femaleOnly
  - **StudySpot**
    - Each studySpot has a boolean to indicate if it's quiet
    - Each studySpot has a boolean to indicate if there are collaborative areas

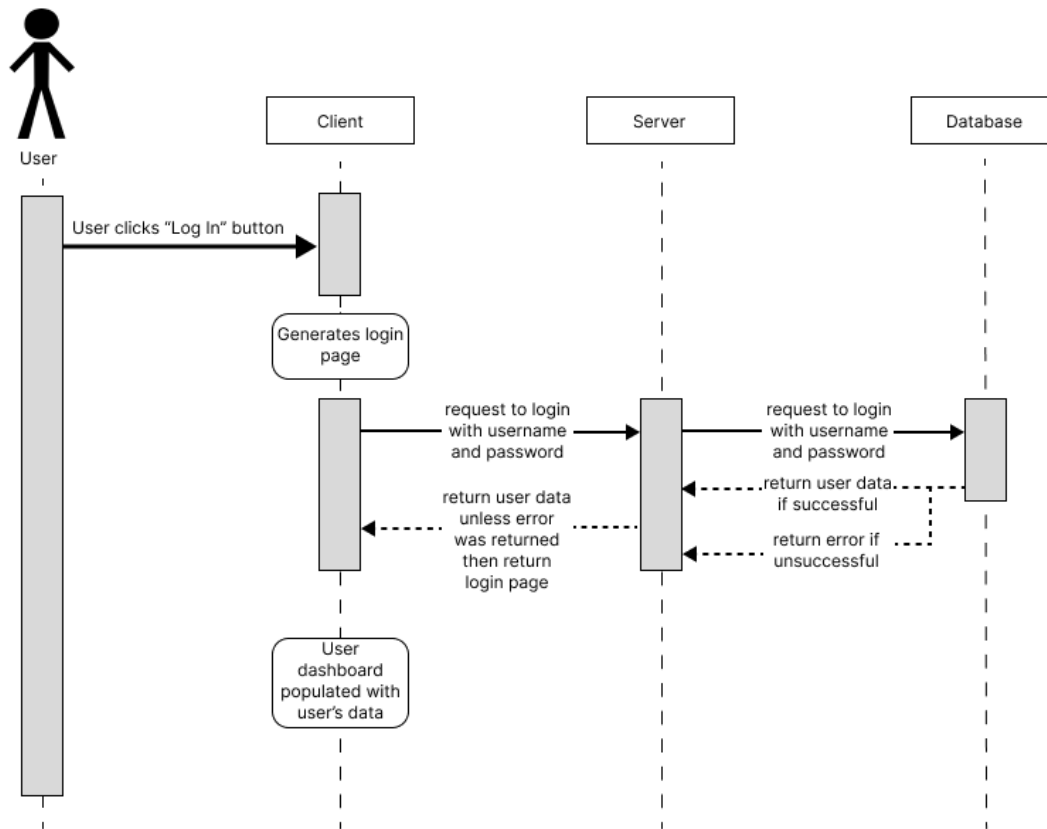
## Sequence Diagram

Sequence of events of how a user will be interacting with the client and how the client/server/database interacts with one another.

### Sequence of events when the user first starts the application

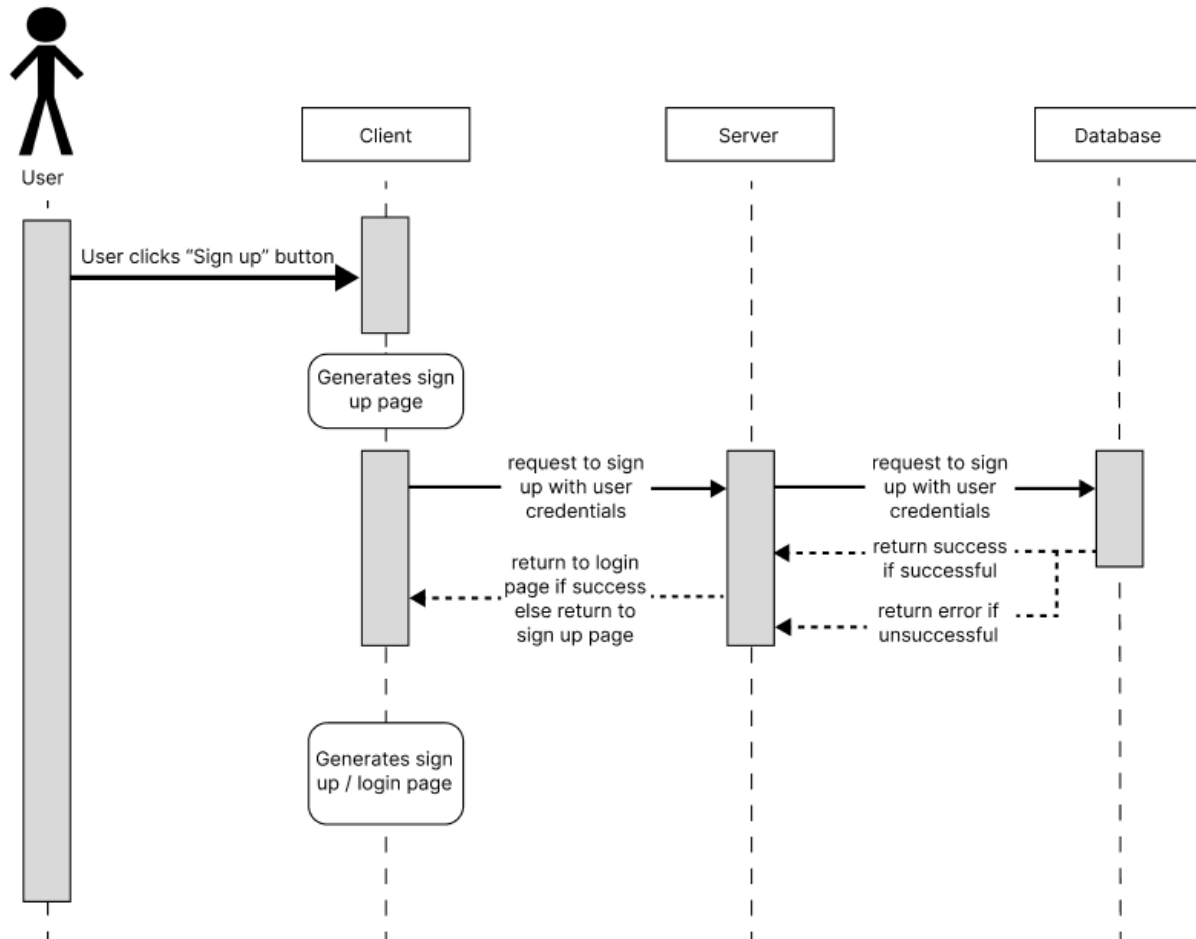


## Sequence of events when a user tries to log in

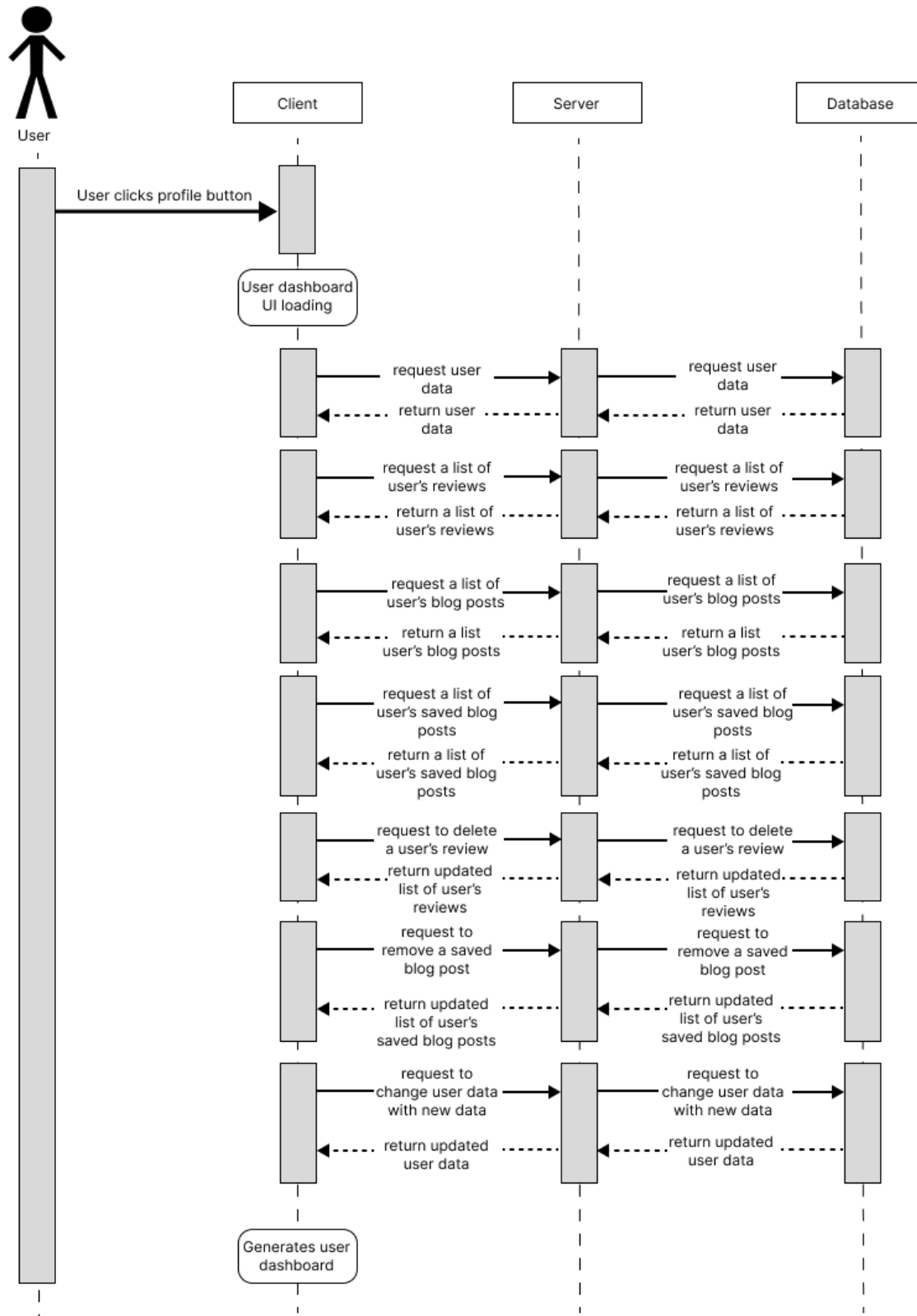




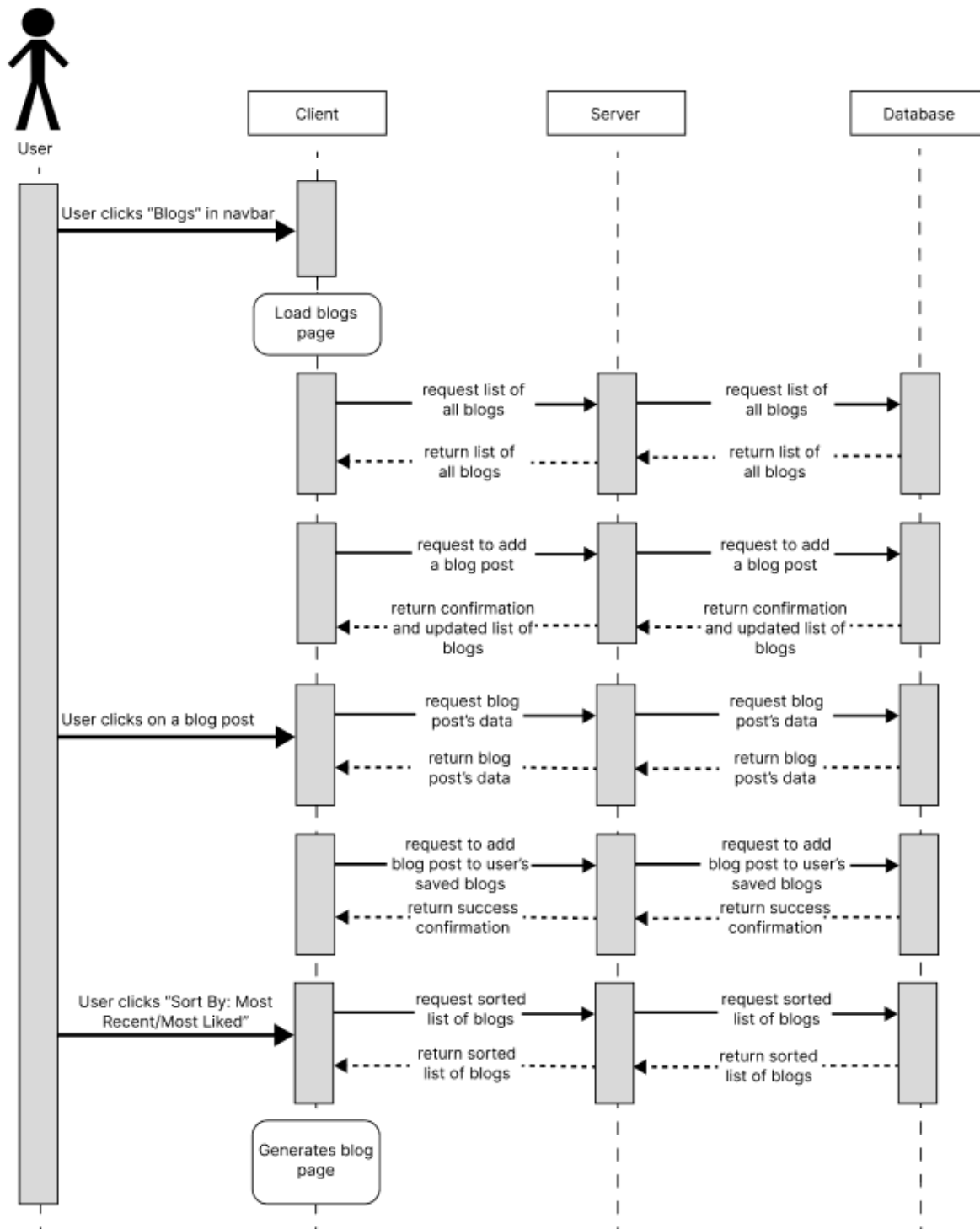
## Sequence of events when a user tries to sign up



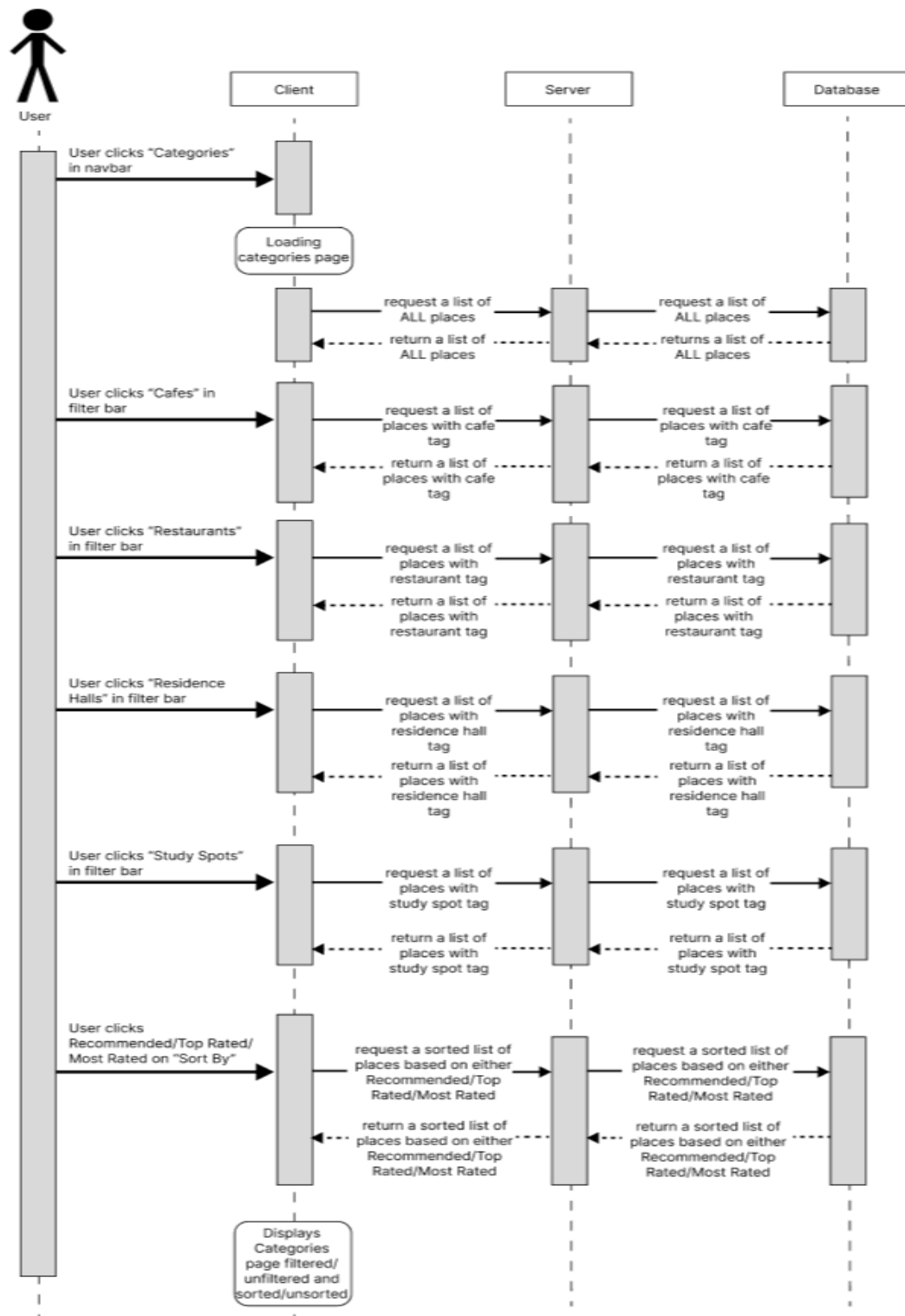
## Sequence of events when a user goes to the user dashboard

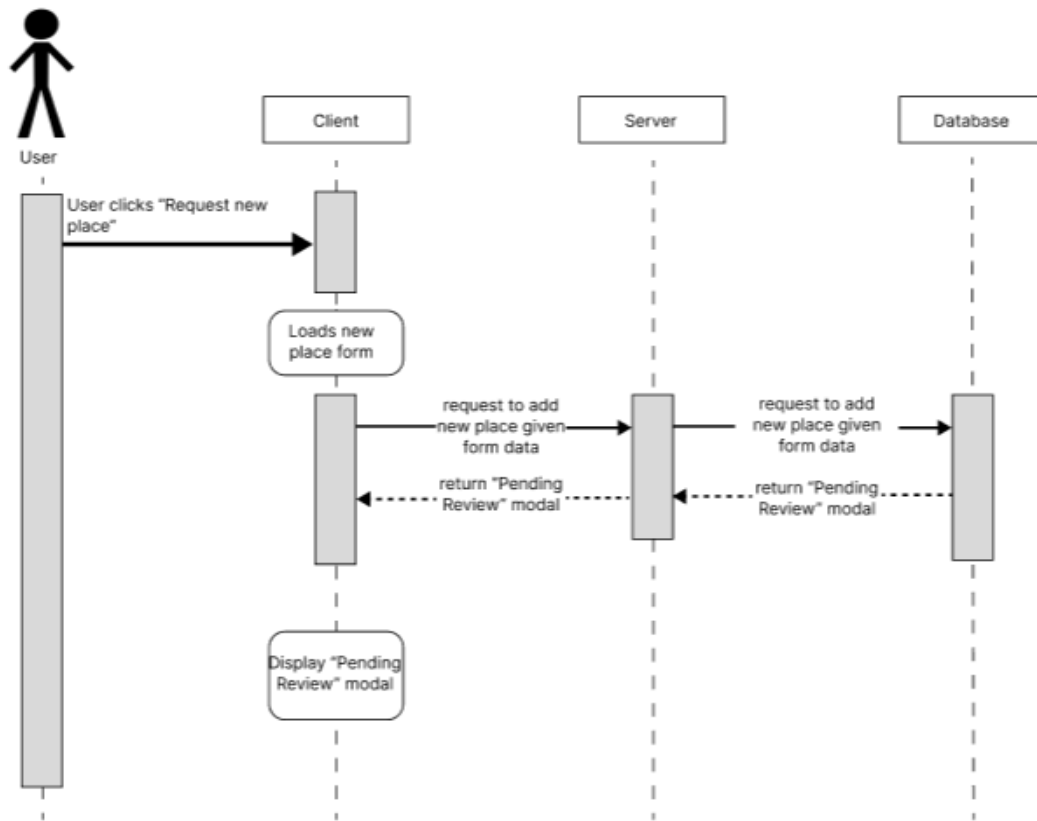


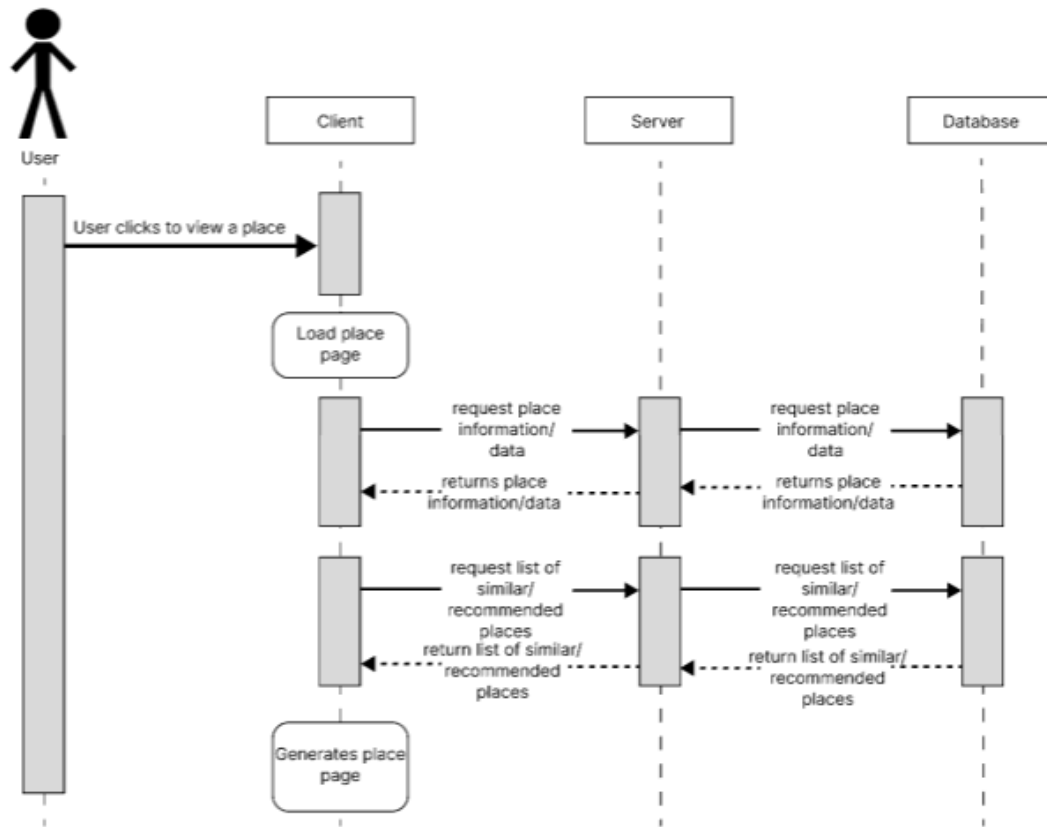
## Sequence of events when user goes to “Blogs” page



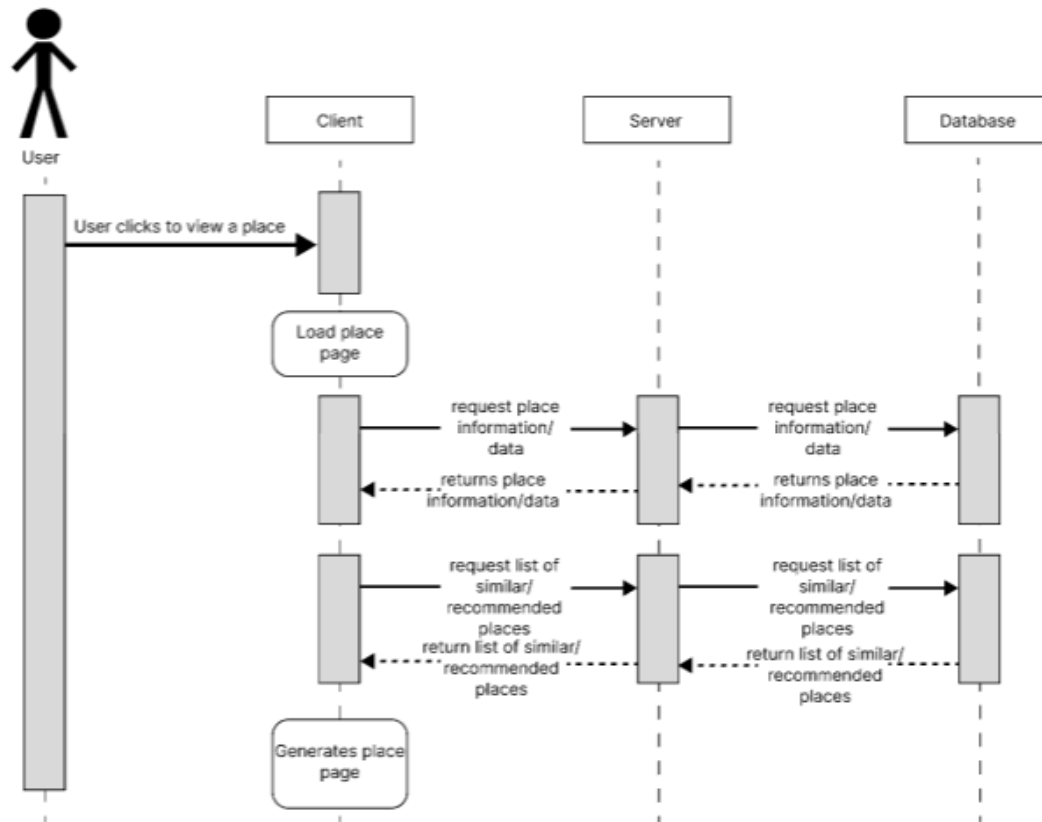
## Sequence of events when the user goes to “Categories” page

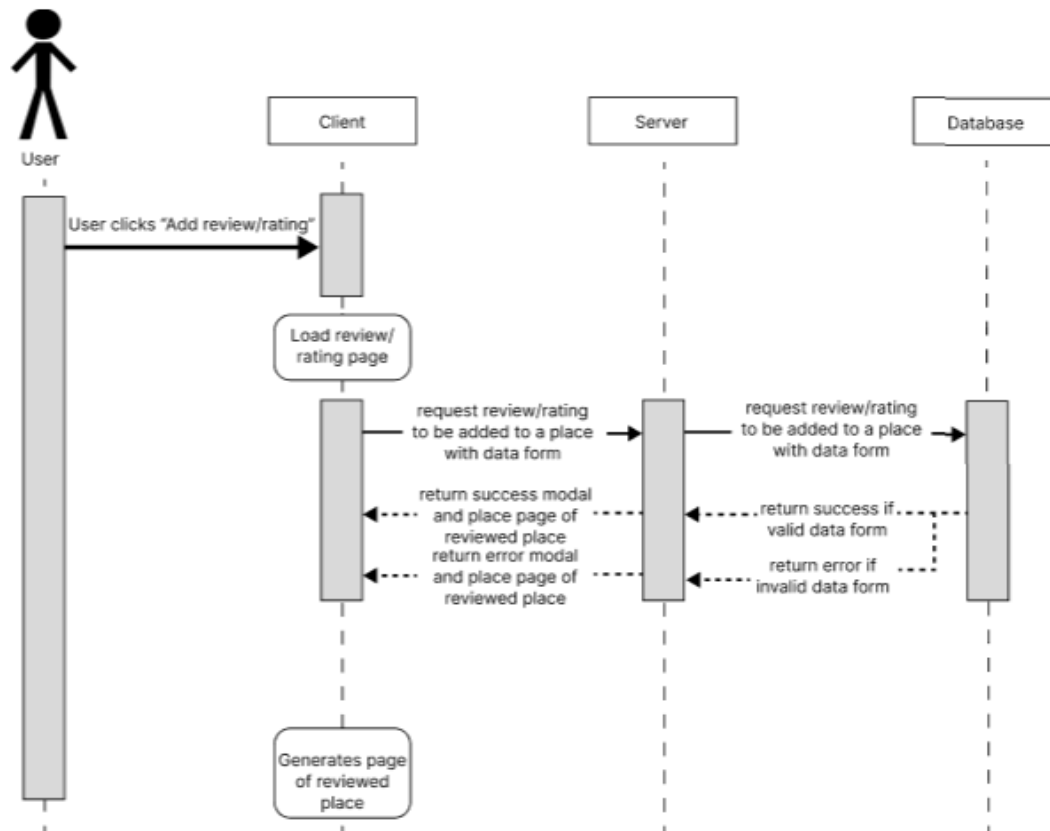


**Sequence of events for a user to request to add a place**

**Sequence of events for a user to request to add a place**

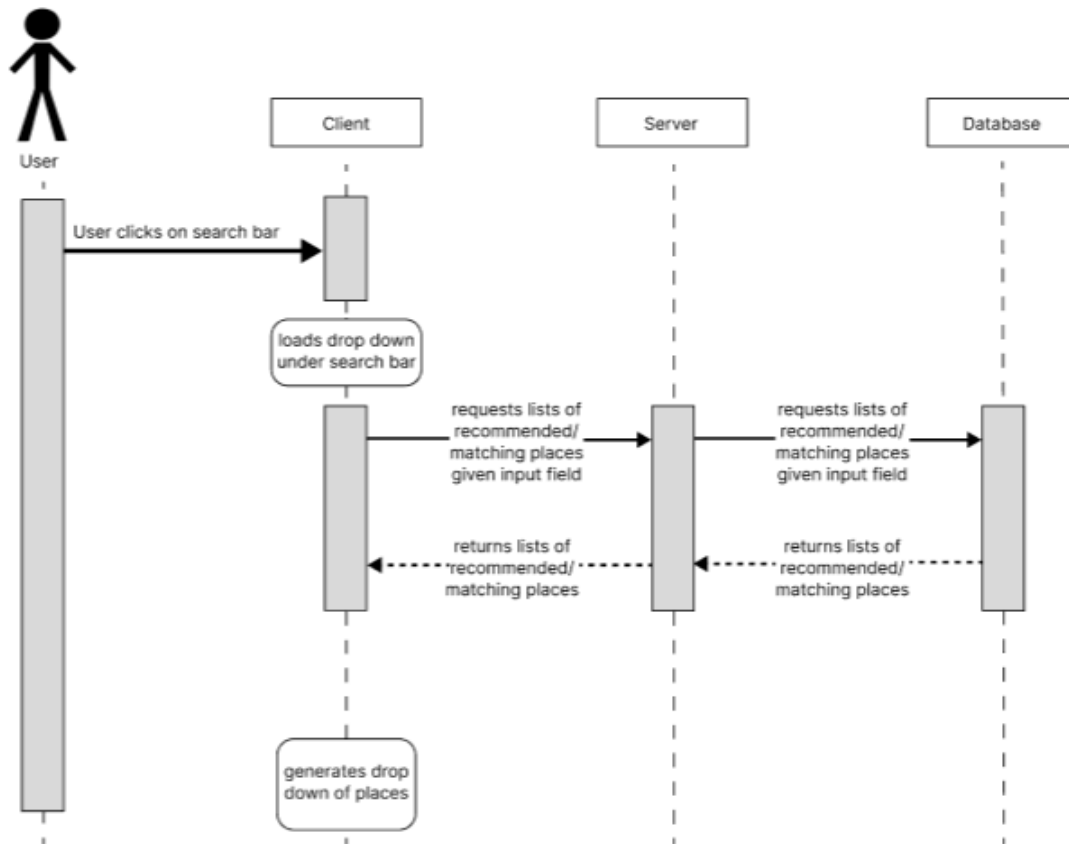
## Sequence of events when a user views a place



**Sequence of events when a user adds a review/rating for a place**

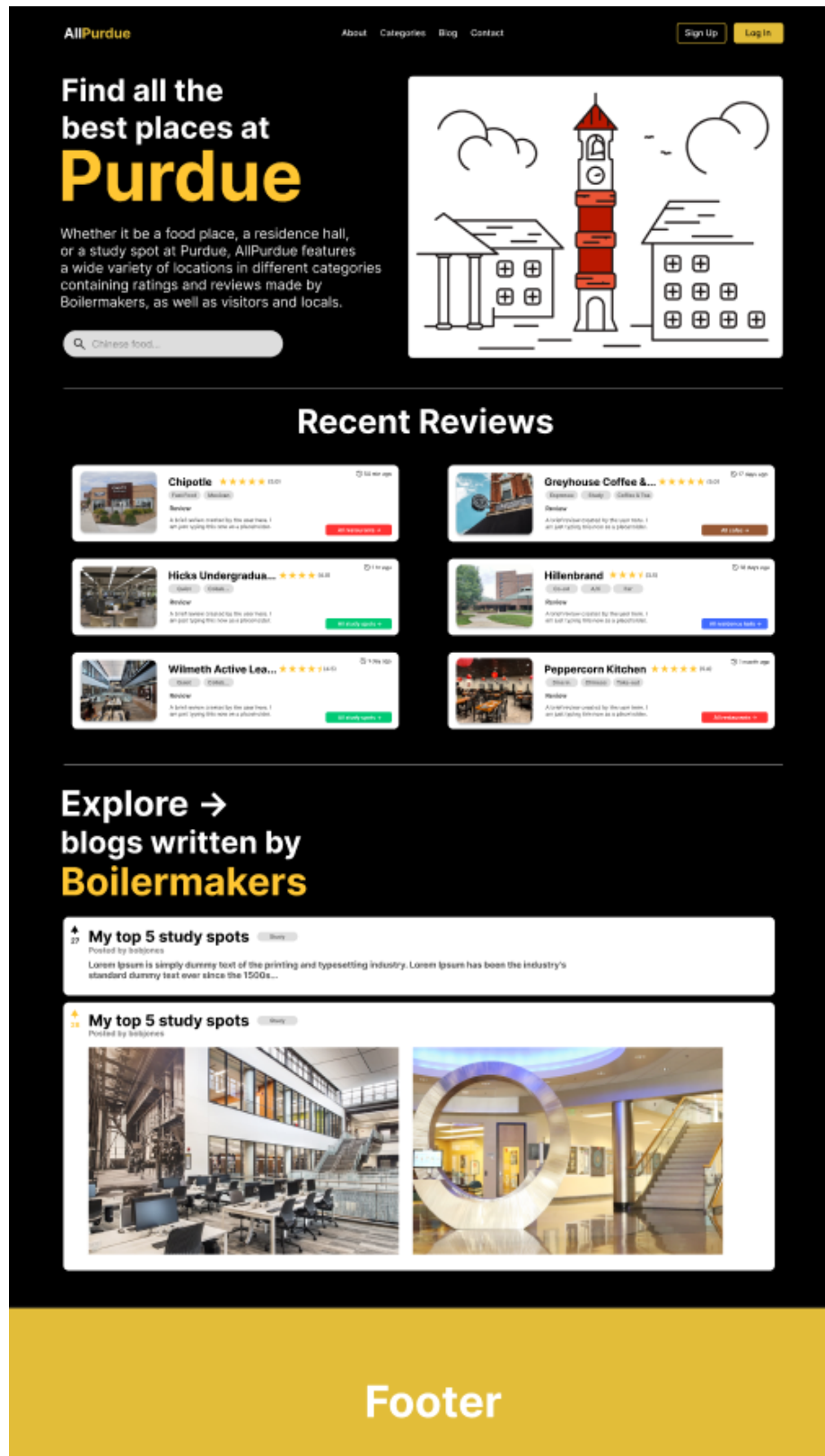


## Sequence of events when a user searches for a place



## UI Mockup

### Home/landing page




## Categories Page (Default)

**AllPurdue**[About](#)[Categories](#)[Blog](#)[Contact](#)[Sign Up](#)[Log In](#)


# Browse All Categories

[Cafes](#)[Restaurants](#)[Residence Halls](#)[Study Spots](#)


**All Places****Sort: Recommended ▾**




**Greyhouse Coffee & Supply Co.**  
★★★★★ 31  
[Expresso](#) [Study](#) [Coffee & Tea](#)  
Classic coffeehouse with a homey vibe & a menu of baked goods, crepes, gelato & smoothies.  
[All cafes →](#)



**Peppercorn Kitchen**  
★★★★★ 46  
[Chinese](#) [Dine-in](#) [Takeout](#)  
PEPPERCORN'S KITCHEN offers the famous "ma la" flavor, literally "numbing and spicy". Perfectly blended in with the Western culture, our large selection of delicacies impresses our guests [more →](#)  
[All restaurants →](#)



**Hick's Undergraduate Library**  
★★★★★ 17  
[Quiet](#) [Collaborative](#) [Study Spot](#)  
Since it opened in 1982, the goal of the Hick's Undergraduate Library has been to create an active learning environment for the Purdue community.  
[All study spots →](#)



**Chipotle**  
★★★★★ 8  
[Mexican](#) [Fast Food](#)  
Chipotle was born of the radical belief that there is a connection between how food is raised and prepared, and how it tastes. Real is better. Better for You. Better for People. Better for Our [more →](#)  
[All restaurants →](#)


## Categories Page (Filtered)

**AllPurdue**[About](#)[Categories](#)[Blog](#)[Contact](#)[Sign Up](#)[Log In](#)

# Browse All Categories

[Cafes](#)[Restaurants](#)[Residence Halls](#)[Study Spots](#)


**All Places > Restaurants****Sort: Recommended ▾**



**Chipotle**  
★★★★★ 8  
[Mexican](#) [Fast Food](#)

Chipotle was born of the radical belief that there is a connection between how food is raised and prepared, and how it tastes. Real is better. Better for You, Better for People, Better for Our [more](#) →

[All restaurants ▾](#)



**Peppercorn Kitchen**  
★★★★★ 45  
[Chinese](#) [Take-out](#) [Takeout](#)

PEPPERCORNS KITCHEN offers the famous "ma la" flavor, literally "numbing and spicy". Perfectly blended in with the Western culture, our large selection of delicacies impresses our guests [more](#) →

[All restaurants ▾](#)

AllPurdue

AboutCategoriesBlogContact

< Blogs

↑  
28

My top 5 study spots

Study

Posted by bobjones · 14 hours ago

2 Comments

Share

Save

Comment as billybob

What are your thoughts?

Comments

Sort By: Recent ▾

sam328 · 9 hr. ago

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English.

joe42 · 12 hr. ago

Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).